7 Some numerical methods

7.1 Numerical solution to Bellman equations

When we cannot solve the Bellman equation analytically, there are several methods to approximate a solution numerically. One of the most straightforward methods when the problem is autonomous, is to discretize the state space and iterate on the Bellman equation until it converges. When the Bellman equation is a contraction mapping, strong results make sure that this procedure converges to the correct value function.

When we discretize the state space, we restrict the state variable to take values from a finite set

$$k_t \in \left[k^1, k^2, ...k^n\right] \equiv K \tag{1}$$

where the superscripts index the elements of K. We then solve for c from the law-of-motion for k

$$\begin{aligned} k_{t+1} &= f\left(k_t, c_t\right) && (2) \\ &\implies c_t = \bar{f}\left(k_t, k_{t+1}\right) && (3) \end{aligned}$$

We can then write the Bellman equation for the discretized problem as

$$V\left(k_t\right) = \max_{k_{t+1} \in K} u\left(k_t, \bar{f}\left(k_t, k_{t+1}\right)\right) + \beta V\left(k_{t+1}\right). \tag{4}$$

As you see, this is a Bellman for a constrained problem, i.e., the control variable is constrained relative to the case when k_{t+1} is continuous. Two things should be noted; First the Bellaan equation is the true Bellman equation of the constrained problem and previous results hold, in particular the contraction mapping theorems apply. Second, how important the constraint implied by the discretization depend on how fine grid it implies. Many (few) elements of K with small (large) distances between them, imply that the constraint is weak (severe).

Denoting an arbitrary initial value function by $V_0\left(k_t\right)$, being n numbers, we update this value function according to

$$V_1\left(k_t\right) = \max_{k_{t+1} \in K} u\left(k_t, \bar{f}\left(k_t, k_{t+1}\right) + \beta V_0\left(k_{t+1}\right)\right) \tag{5}$$

giving $V_1\left(k_t\right)$, being a new set of n numbers. We then iterate on the Bellman equation

$$V_{s+1}\left(k_t\right) = \max_{k_{t+1} \in K} u\left(k_t, \bar{f}\left(k_t, k_{t+1}\right) + \beta V_s\left(k_{t+1}\right)\right) \tag{6}$$

until $V_{s+1}(k_t) \approx V_s(k_t)$. For each of the n values of k_t, we check $u\left(k_t, \bar{f}(k_t, k_{t+1})\right) + \beta V(k_{t+1})$ for all n values of k_{t+1} and choose the k_{t+1} that gives the highest value, giving $V_{s+1}(k_t)$. Therefore, each iteration requires n^2 evaluations when the state variable is unidimensional.[7]

Lets consider a simple example.

$$\begin{aligned} & \max_{\{c_t\}_t^\infty} \sum_{t=0}^{\infty} \beta^t \ln(c_t) && (7) \\ s.t. k_{t+1} &= f(k_t, c_t) \equiv k_t^\alpha + (1-\delta) k_t - c_t && (8) \\ k_t &\geq 0 \forall t && (9) \\ k_0 &= \underline{k} && (10) \end{aligned}$$

First, we solve for

$$c_t = \bar{f}(k_t, k_{t+1}) = k_t^\alpha + (1-\delta) k_t - k_{t+1}. \quad (11)$$

Then, we note that $k_t \in \left[0, \delta^{\frac{1}{\alpha-1}}\right] \implies k_{t+1} \in \left[0, \delta^{\frac{1}{\alpha-1}}\right]$, implying that value function is bounded. If also $\beta < 1$, the Bellman equation

$$V(k_t) = \max_{c_t} \ln(k_t^\alpha + (1-\delta) k_t - k_{t+1}) + \beta V(k_{t+1}) \quad (12)$$

is a contraction mapping.

Let us parametrize, setting $\beta = 0.9$, $\delta = .2$ and $\alpha = 1/2 \implies \delta^{\frac{1}{\alpha-1}} = 25$ and discretize the state space by requiring

$$k_t \in [5, 10, 15, 20, 25] \equiv K \forall t. \quad (13)$$

Now set an initial value function, for example

$$V_0(k) = \ln k \ \forall k.$$

This is then updated in the following way. For each possible k_t, k_{t+1} we calculate the left hand side of the Bellman equation, and solve the maximization problem So, for $k_t = 5$, and all $k_{t+1} \in K$ we have

$$\begin{aligned} \ln(5^\alpha + (1-\delta) 5 - 5) + .9 \ln 5 &= 1.66 \\ \ln(5^\alpha + (1-\delta) 5 - 10) + .9 \ln 10 &= -\infty \\ \ln(5^\alpha + (1-\delta) 5 - 15) + .9 \ln 15 &= -\infty \\ \ln(5^\alpha + (1-\delta) 5 - 20) + .9 \ln 20 &= -\infty \\ \ln(5^\alpha + (1-\delta) 5 - 25) + .9 \ln 25 &= -\infty \end{aligned}$$

[7]When the state variable is higher dimensionality, this method quickly become to computationally burdensome.

Implying that the updated value function for $k_t = 5$ is

$$V_1(5) = 1.66.$$

For $k_t = 10$,

$$\begin{aligned}
\ln\left(10^\alpha + (1-\delta)\,10 - 5\right) + .9\ln 5 &= 3.27 \\
\ln\left(10^\alpha + (1-\delta)\,10 - 10\right) + .9\ln 10 &= 2.22 \\
\ln\left(10^\alpha + (1-\delta)\,10 - 15\right) + .9\ln 15 &= -\infty \\
\ln\left(10^\alpha + (1-\delta)\,10 - 20\right) + .9\ln 20 &= -\infty \\
\ln\left(10^\alpha + (1-\delta)\,10 - 25\right) + .9\ln 25 &= -\infty
\end{aligned}$$

implying

$$V_1(10) = 3.27$$

In the same way, for $k_t = 15$

$$\begin{aligned}
\ln\left(15^\alpha + (1-\delta)\,15 - 5\right) + .9\ln 5 &= 3.83 \\
\ln\left(15^\alpha + (1-\delta)\,15 - 10\right) + .9\ln 10 &= 3.84 \\
\ln\left(15^\alpha + (1-\delta)\,15 - 15\right) + .9\ln 15 &= 2.30 \\
\ln\left(15^\alpha + (1-\delta)\,15 - 20\right) + .9\ln 20 &= -\infty \\
\ln\left(15^\alpha + (1-\delta)\,15 - 5\right) + .9\ln 25 &= -\infty
\end{aligned}$$

$$V_1(15) = 3.84$$

Doing this also for $k_t = 20$ and $k_t = 25$ completes the first iteration. Then, we repeat the iterations until we think the process has converged sufficiently,

7.2 Band Matrix Methods for differential equations

Assume we want to solve the differential equation

$$y''(t) + ay'(t) + by(t) = g(t) \tag{14}$$

over some interval, with initial conditions given. If we want to solve this numerically, we first have to get rid of the abstract infinitely small differences called differentials. We approximate these with finite size forward differences such that

$$y'(t) \approx \frac{y(t+\Delta t) - y(t)}{\Delta t}, \tag{15}$$

$$y'(t) \approx \frac{\frac{y(t+\Delta t)-y(t)}{\Delta t} - \frac{y(t)-y(t-\Delta t)}{\Delta t}}{\Delta t} \tag{16}$$

$$= \frac{y(t+\Delta t) - 2y(t) + y(t-\Delta t)}{\Delta t^2} \tag{17}$$

Using this we can solve the equation for a finite set of values in the following way. Say we want to solve the equation in the interval $t \in [p, q]$ and we know that $y'(p) = y''(p) = 0$. We divide the interval for t into n equal parts and use the following notation

$$\begin{aligned} t_k &= p + \frac{k(q-p)}{n}, & (18) \\ t_k - t_{k-1} &= \frac{q-p}{n} \equiv \Delta t, & (19) \\ y(t_k) &\equiv y_k. & (20) \end{aligned}$$

This gives us the following equations

$$\begin{aligned} \frac{(y_{-1} - 2y_0 + y_1)}{\Delta t^2} &= 0 & (21) \\ \frac{(-y_0 + y_1)}{\Delta t} &= 0 & (22) \\ \frac{(y_{-1} - 2y_0 + y_1)}{\Delta t^2} + a\frac{-y_0 + y_1}{\Delta t} + by_0 &= g(t_0) & (23) \\ \frac{(y_0 - 2y_1 + y_2)}{\Delta t^2} + a\frac{-y_1 + y_2}{\Delta t} + by_1 &= g(t_1) & (24) \\ &. & (25) \\ &. & (26) \\ \frac{(y_{n-1} - 2y_n + y_{n+1})}{\Delta t^2} + a\frac{-y_n + y_{n+1}}{\Delta t} + by_n &= g(t_n). & (27) \end{aligned}$$

This provides $n + 3$ linear equations for the $n + 3$ unknown y. Writing this as a system we have

$$\mathbf{A} \begin{bmatrix} y_{-1} \\ y_0 \\ y_1 \\ . \\ y_n \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ g(t_0) \\ g(t_1) \\ . \\ g(t_n) \end{bmatrix} \qquad (28)$$

$$\begin{bmatrix} y_{-1} \\ y_0 \\ y_1 \\ . \\ y_n \\ y_{n+1} \end{bmatrix} = \mathbf{A}^{-1} \begin{bmatrix} 0 \\ 0 \\ g(t_0) \\ g(t_1) \\ . \\ g(t_n) \end{bmatrix}, \qquad (29)$$

with (setting $n = 3$)

$$\mathbf{A} \equiv \tag{30}$$

$$\begin{bmatrix} \Delta t^{-2} & -2\Delta t^{-2} & \Delta t^{-2} & 0 & 0 & 0 \\ 0 & -\Delta t^{-1} & \Delta t^{-1} & 0 & 0 & 0 \\ \Delta t^{-2} & -2\Delta t^{-2} - a\Delta t^{-1} + b & \Delta t^{-2} + a\Delta t^{-1} & 0 & 0 & 0 \\ 0 & \Delta t^{-2} & -2\Delta t^{-2} - a\Delta t^{-1} + b & \Delta t^{-2} + a\Delta t^{-1} & 0 & 0 \\ 0 & 0 & \Delta t^{-2} & -2\Delta t^{-2} - a\Delta t^{-1} + b & \Delta t^{-2} + a\Delta t^{-1} & 0 \\ 0 & 0 & . & \Delta t^{-2} & -2\Delta t^{-2} - a\Delta t^{-1} + b & \Delta t^{-2} + a\Delta t^{-1} \end{bmatrix}$$

To get any accuracy, we should of course set n much larger than 3. As we see, the matrix $\mathbf{A}$ contains many zeros, with a *band* of non-zeros around the diagonal. Due to this feature, it is easy for the computer to invert it also if n is in he order of hundreds.

7.3 Newton-Raphson

Suppose we are looking for an optimum of the real valued function $f(\mathbf{x}), \mathbf{x} \in \mathbf{R}^n$ where f is twice differentiable. A standard way to do this numerically is to apply the Newton-Raphson algorithm. If the optimum is interior, it satisfies the necessary first order conditions that the gradient is zero, i.e., at the optimum, denoted $\mathbf{x}^*$,

$$Df(\mathbf{x}) = 0. \tag{31}$$

Now apply a first order linear approximation to the gradient from some initial point $\mathbf{x}_0$

$$0 = Df(\mathbf{x}^*) \approx Df(\mathbf{x}_0) + D^2 f(\mathbf{x}_0)(\mathbf{x}^* - \mathbf{x}_0), \tag{32}$$

where $D^2 f(\mathbf{x}_0)$ is the *Hessian* matrix of second derivatives of f.

Provided the Hessian is invertible, we can get an approximation to $\mathbf{x}^*$,

$$\mathbf{x}^* \approx \mathbf{x}_0 - \left(D^2 f(\mathbf{x}_0)\right)^{-1} Df(\mathbf{x}_0). \tag{33}$$

From this we can construct a search algorithm that under some circumstances makes better and better approximations

$$\mathbf{x}_{s+1} \approx \mathbf{x}_s - \left(D^2 f(\mathbf{x}_s)\right)^{-1} Df(\mathbf{x}_s). \tag{34}$$

If we don't have analytic solutions to the gradient and Hessian we can use numerical approximations, for example the forward difference method;

For a small number ε, we have,

$$\frac{\partial f(\mathbf{x})}{\partial x_1} = \frac{f\left(\mathbf{x}+\left[\begin{array}{c}\varepsilon\\ \mathbf{0}\end{array}\right]\right)-f(\mathbf{x})}{\varepsilon} \tag{35}$$

$$\frac{\partial f(\mathbf{x})}{\partial x_2 \partial x_1} = \frac{\frac{\partial f\left(\mathbf{x}+\left[\begin{array}{c}0\\ \varepsilon\\ \mathbf{0}\end{array}\right]\right)}{\partial x_1}-\frac{\partial f(\mathbf{x})}{\partial x_1}}{\varepsilon}. \tag{36}$$

One should be very careful with this method since it can only find local optima in the case when f is not globally concave. In well-behaved problems, it is however, easily programmed and fairly quick.